**Lucas Miguel de Matos Negri – RA: 19.00386-2**

**SIMULADO**

1)a) PREORDEM: 78, 13, 12, 99, 37, 81, 65, 29, 13

1)b) POSORDEM: 81, 65, 12, 37, 29, 13, 13, 99, 78

2) profundidade 1

3) altura 1

5) 1023 nós

6) altura 10

7)a)

99 (Mauro) -> pai

35 (Silvio) -> filho de 99

15 (Ana) -> filho de 35

3 (Paula) -> filho de 15

11 (Selma) -> filho de 15

50 (Leandro) -> filho de 35

24 (Bia) -> filho de 50

16 (Selma) -> filho de 24

36 (Teo) -> filho de 50

25 (Carlos) -> filho de 36

37 (Camila) -> filho de 25

57 (Jorge) -> filho de 35

100 (Sueli) -> filho de 99

124 (Artur) -> filho de 100

65 (Mark) -> filho de 100

28 (Ivo) -> filho de 99

7)b)

Comece na raiz com 99 (Mauro).

Compare 25 com 99. Como 25 é menor que 99, vá para o filho à esquerda.

Agora você está em 35 (Silvio).

Compare 25 com 35. Como 25 é menor que 35, vá para o filho à esquerda.

Agora você está em 15 (Ana).

Compare 25 com 15. Como 25 é maior que 15, vá para o filho à direita.

Agora você está em 24 (Paula).

Compare 25 com 24. Como 25 é maior que 24, vá para o filho à direita.

Agora você encontrou 25 (Carlos), que é a chave desejada.

Portanto, as comparações de chaves processadas para buscar a chave 25 são: 99 (raiz) -> 35 -> 15 -> 24 -> 25.

8) D) 93 92 78 60 28 66 70 39 19.

9) A sequência 33 32 28 31 26 29 25 30 27 não é um heap válido, pois não segue as propriedades de um heap máximo. Alguns dos elementos pais são menores que seus filhos, violando a definição de um heap máximo. Portanto, essa sequência não é um heap máximo válido.

10)a)   
public class Node\_Tree {

public static void ImprimeQtdeFilhos(Node\_Int node) {

if (node.firstChild == null) {

System.out.println("Nó sem filhos...");

} else {

int count = 0;

Node\_Int child = node.firstChild;

while (child != null) {

count++;

child = child.next;

}

System.out.println("Quantidade de filhos: " + count);

}

}

public static void main(String[] args) {

// Exemplo de uso

Node\_Int A = new Node\_Int(1);

Node\_Int B = new Node\_Int(2);

Node\_Int C = new Node\_Int(3);

Node\_Int D = new Node\_Int(4);

A.firstChild = B;

B.next = C;

C.next = D;

ImprimeQtdeFilhos(A); // Deve imprimir "Quantidade de filhos: 3"

}

}  
  
10)b)

public class Node\_Tree {

public static void DobraValoresFilhos(Node\_Int node) {

if (node.firstChild == null) {

System.out.println("Nó sem filhos...");

} else {

Node\_Int child = node.firstChild;

while (child != null) {

child.item \*= 2;

child = child.next;

}

}

}

public static void main(String[] args) {

// Exemplo de uso

Node\_Int A = new Node\_Int(1);

Node\_Int B = new Node\_Int(2);

Node\_Int C = new Node\_Int(3);

Node\_Int D = new Node\_Int(4);

A.firstChild = B;

B.next = C;

C.next = D;

DobraValoresFilhos(A);

// Valores dos filhos de A são agora: 2, 6, 8

}

}